**Chapter 15: Build and Deploy Techniques**

Add a note hereIf you’ve read the preceding chapters, you are now familiar with some of the capabilities of the DataPower device. You should now understand the techniques required to configure XML Firewall services from the device’s WebGUI interface. You might have also read [Chapter 13](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=2191#2191), [“Alternate Management Interfaces”](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=2191#2191) and discovered other configuration methodologies such as the command-line interface (CLI) and the XML Management Interface and found that these facilities, while typically not primary service configuration tools, can be used for the modification and ongoing maintenance of services.

Add a note hereThis chapter takes a look at DataPower configuration from another angle. In this chapter, we look at the details of how configurations are defined and managed on the device, and we look at the migration of configurations across the spectrum of environments—from initial coding to testing, acceptance, and production.

**Add a note here****Goals and Objectives of Configuration, Build, and Deployment**

Add a note hereIn order to meet our objectives, we present configuration methods, including WebGUI, CLI, and XML Management Interface. We describe the use of DataPower configuration options for migration issue mediation, including externalizing static data, and the modification of configuration parameters. Techniques that ensure that DataPower devices are managed in a consistent and controlled manner in a multi-device and highly available configuration are also presented. This investigation leads to a sample configuration architecture and a discussion of the migration of configuration from development through production.

Add a note hereFinally, the use of supporting products such as IBM Tivoli Composite Application Manager System Edition for DataPower (ITCAMSEDP) and the Integrated Solutions Console (ISC) are introduced as facilitators of multi-device management platforms.

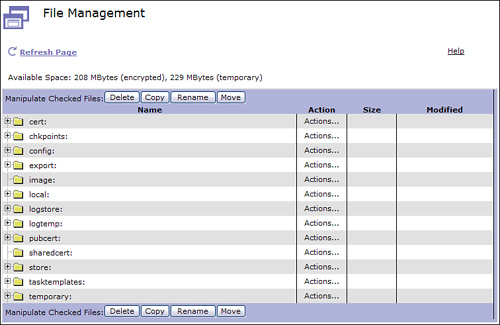
**Add a note here****DataPower Configuration Fundamentals**

Add a note hereBefore we get too deep into our discussion, let’s briefly review the associated topics that will play into the methodologies we demonstrate. Some of these topics are addressed in [Chapter 12](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=2003#2003), [“Device Administration,”](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=2003#2003) so we’ll just cover enough to refresh your memory.

**File System**

Add a note hereThe DataPower file system is an encrypted data source separated into several named directories. Directories are used to host configuration data, store XSLT stylesheets, capture logging events, manage cryptographic certificates and keys, and control other system functions.

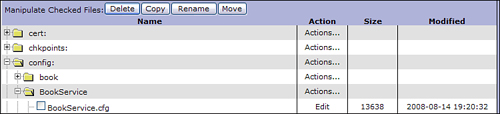
Add a note hereEach device contains one or more configuration files that describe the details of all services and objects. We look closely at these files shortly. Configuration files are stored in the config: directory, while custom data maintained by the user is stored in the local: directory. The device stores most of its required files in a system directory called store:. You’ll find a complete description of the file system in the WebGUI Guide for your device. Figure 15-1 shows a typical directory structure.

[](javascript:PopImage('IMG_332','http://images.books24x7.com/bookimages/id_30903/15fig01_alt.jpg','756','491'))  
Add a note hereFigure 15-1: DataPower file system.

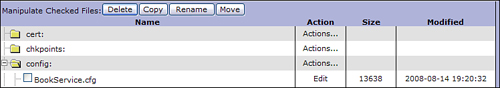
**Application Domains**

Add a note hereThe DataPower file system is initially booted with a single domain labeled “default.” The file system may be further segmented with the creation of application domains. You can read about domains in [Chapter 12](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=2003#2003); however, for our discussion you should know that a newly created domain is provided with its own local: and config: directories and will be configured for read access to files in the store: directory of the default domain. Access to other application domains may also be granted.

Add a note hereSubdirectories may be created. For example, each application domain’s configuration is hosted in the default domain in a file (with .cfg suffix) and subdirectory of the same name. Figure 15-2 shows the book and BookService subdirectories, and the BookService.cfg configuration file.

[](javascript:PopImage('IMG_333','http://images.books24x7.com/bookimages/id_30903/15fig02_alt.jpg','750','171'))  
Add a note hereFigure 15-2: Config subdirectory from within default domain.

Add a note hereThe BookService subdirectory is also accessible from within the application domain itself, without the default domain hierarchy. Figure 15-3 shows the same configuration file (BookService.cfg), now accessible from within the config: directory of the BookService domain.

[](javascript:PopImage('IMG_334','http://images.books24x7.com/bookimages/id_30903/15fig03_alt.jpg','748','131'))  
Add a note hereFigure 15-3: Config subdirectory from within application domain.

**Devices and Environments**

Add a note hereEach DataPower device may participate as a member of a peer group that provides services and shares management information such as Service Level Management (SLM) data with other members of the group. For the purpose of our discussion, a shared environment of devices refers to the group of devices servicing Software Development Life Cycle (SDLC) areas such as development, test, acceptance, or production. Migrating configuration information to an environment refers to the entire group of devices in that environment.

Add a note hereUsing a single device to host multiple SDLC environments such as test and acceptance complicates SDLC migration as certain objects such as DNS Static Hosts (which we will discuss later) are defined at the device level and sharing them across life cycle environments breaks their functionality.

**Tip: Avoid Using a Device for Multiple Application Life Cycle Environments**

Add a note hereUsing a single device for more than one SDLC environment is unwise. Sharing a production device is foolhardy and risks disrupting production activities.

**Load Balancers**

Add a note hereIt is often the case that groups of devices, or perhaps an entire environment, will reside behind a load balancer that acts as a façade, exposing a single Virtual IP Address (VIP) for request traffic. This topology has no effect on the configuration of the device. There will be no device affinity (unless it has been specified at the load balancer itself) as each transaction can be processed by any of the devices in the load balanced group. Service level data is shared by devices via peer group registration, and all transactions participate in service level management. You can read more about SLM in [Chapter 10](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=1548#1548), [“Web Service Proxy.”](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=1548#1548)

**Configuration Persistence**

Add a note hereRegardless of which configuration method was used to create DataPower objects, the end result is an entry (in CLI format) in an onboard configuration file within the device’s config: directory. Listing 15-1 shows the result of creating an application domain using any of the configuration methods as it is written to the configuration file.

Add a note hereListing 15-1: Application Domain Configuration Detail

Add a note heredomain "BookService"

base-dir BookService:

base-dir local:

config-file BookService.cfg

visible-domain default

url-permissions http+https+snmp+ftp+mailto+mq

file-permissions CopyFrom+CopyTo+Delete+Display+Exec+Subdir

config-mode import

import-url "http://192.168.1.35:9099/BookServiceExport.zip"

exit

Add a note hereConfiguration changes are not persisted until a save configuration action is invoked, either from the WebGUI’s Save Config link, the CLI write memory command, or XML Management’s SaveConfig action. Changes may be saved for a single domain or all domains. Until this is done, the changes are only part of the running configuration.

Add a note hereWhen the save action is initiated, the default domain configuration is persisted in a file named autoconfig.cfg, while application domain configuration is persisted in the configuration file defined in the application domain object. This name defaults (as has been seen in the previous file system discussion) to the name of the domain itself.

Add a note hereYou may have more than one default domain configuration file on the device at a time, although only one is the active configuration. This allows for the device to be booted into one of several different boot profiles depending on your needs. For example, on a testing device, you might have configuration files for several different services and boot the device with one or the other depending on your testing needs. Figure 15-4 shows the ability to select which configuration file to boot the device with. This is available from the Administration→System Control screen and is only available in the default domain. Restarting the device utilizes the selected configuration.

[](javascript:PopImage('IMG_335','http://images.books24x7.com/bookimages/id_30903/15fig04.jpg','500','92'))  
Add a note hereFigure 15-4: Select configuration showing multiple configuration files.

**Add a note here****Configuring for Migration**

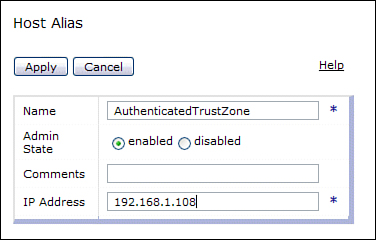
Add a note hereBefore configuring a service, it is important to understand the methods, objects, and properties that will best provide for eventual migration to another environment. You will see how the proper use of network objects such as DNS Servers, DNS Static Hosts, and Host Aliases, and the avoidance of static endpoint information in XSLT will greatly ease migration issues and preclude having to modify configurations to fit the targeted environment.

**Network Objects**

Add a note hereThe fundamental problem encountered during migration is the creation of environmental affinities within device configurations. This is often a result of the designation of off box services via dot decimal IP addresses, or the use of environment-specific DNS names. Other affinities can also be created with non-IP-related properties such as ports, MQ queue names, or channels. The following network objects (defined in the default domain) will assist in avoiding these affinities. You may also want to refer to [Chapter 4](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=403#403), [“Advanced DataPower Networking”](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=403#403) for a more thorough investigation of these subjects.

**Host Alias**

Add a note hereHost Aliases are used to provide abstract names for the Ethernet interfaces on the device. When using services such as the XML Firewall, Web Service Proxy or Multi-Protocol Gateway, where incoming requests may be bound to particular interfaces on the DataPower device, the use of host alias objects in place of the numeric addresses of those interfaces can alleviate migration issues. A host alias is simply a reference to an IP address on an interface of the device. In this fashion requests can also be restricted to a particular interface, furthering the security of the service by restricting access to particular subnets. Figure 15-5 shows the simple definition of a host alias.

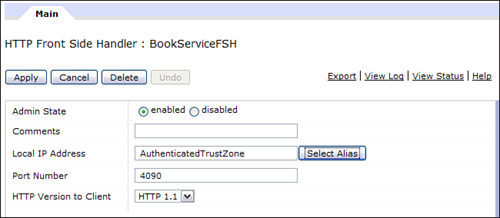
  
Add a note hereFigure 15-5: Host Alias configuration.

**Tip: SDLC Environments and Alias Objects**

Add a note hereIt is recommended that all SDLC environments contain like-named host alias objects. Each environment’s host aliases define addresses specific to its network infrastructure. Furthermore, the host alias names should be self-descriptive, such as those describing the interfaces’ trust characteristics: InternetFacingInterface or AuthenticatedTrustZone.

Add a note hereAs services are migrated, the host aliases are not moved to the target, and as the names are equivalent, the services will automatically utilize the target’s host alias.

Add a note hereAll services can use host aliases. The XML Firewall, for example, uses it on the Front End Device Address definition, while services utilizing Front Side Handler (FSH) objects implement host aliases there. Figure 15-6 shows the association of the host alias with an FSH.

[](javascript:PopImage('IMG_337','http://images.books24x7.com/bookimages/id_30903/15fig06_alt.jpg','650','284'))  
Add a note hereFigure 15-6: FSH with host alias in place of hard-coded interface address.

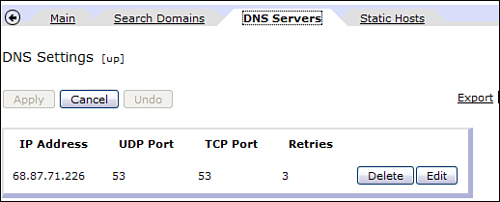
**Tip: Use Hosts Alias Objects Only for IP Addresses**

Add a note hereHost aliases are for interface definitions only! Do not use them for any addresses off the box. Host aliases are selected from the service definition Select Host Alias button and only device addresses are appropriate here.

**DNS**

Add a note hereThe DNS Settings object provides for the designation of device-specific DNS servers and DNS Static Host settings that may be used to extend the DNS services. DNS static hosts provide a powerful aliasing capability that will greatly assist in configuration migration. Using literal names such as highPriorityBookHost.books.net as apposed to a dot decimal address also provides a more streamlined configuration migration.

Add a note hereMost objects such as services, log targets, load balancer group members, and even extension functions, can accept a literal name (DNS or DNS static host) in place of a dot decimal address. Figure 15-7 shows the definition of DNS servers available from the Network→DNS Settings menu.

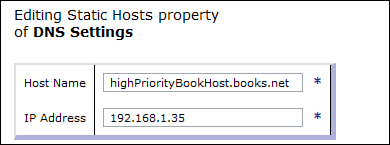
[](javascript:PopImage('IMG_338','http://images.books24x7.com/bookimages/id_30903/15fig07_alt.jpg','518','209'))  
Add a note hereFigure 15-7: DNS Server added to DNS Object.

Add a note hereDNS settings are similar to the host alias objects in that they are defined in the default domain, and are not migrated with services. Therefore, when services move to a target environment, they can immediately utilize the DNS settings defined there.

**DNS Static Hosts**

Add a note hereAlthough the DNS server under most circumstances provides address resolution, DataPower also provides for an additional layer of abstraction. The DNS Static Host might be used to define specific DNS name/address resolution. This is similar to the use of the /etc/hosts file, which also provides an extension of DNS services.

Add a note hereThis technique can be employed as a migration assistance methodology. In this case, DNS static host entries in the production environment would contain different addresses than the test environment DNS static host entries. Figure 15-8 shows the addition of a DNS static host entry to the DNS Settings object.

  
Add a note hereFigure 15-8: DNS Static Host entry added to DNS Object.

**Tip: DNS Static Host Settings**

Add a note hereUse DNS static host settings when environment-specific DNS is not available.

Add a note hereDNS servers should be used as the primary source for address resolution. DNS static host settings may be used for additional address resolution, but caution must be taken to avoid overriding DNS server settings.

**XSLT Issues**

Add a note hereOne area that is often a cause for concern is the use of hard-coded, unnamed numerical constants or *“magic numbers”* in custom XSLT code. Those experienced with DataPower extension functions may be familiar with the possible use of dot decimal addresses or port numbers in routing statements and other extension functions. The use of these hard-coded addresses causes an affinity with the environment. This problem may be easily resolved by externalizing this static data.

Add a note hereThis topic introduces the *identity document.* This is a simple XML Document that can be shared by devices within an environment. It can be resident on the device or fetched from a central location such as a Software Configuration Management (SCM) system. We demonstrate its utility for the externalizing and centralizing of aliased configuration details. The example examines the incoming URL, and when it contains a special character string (Purchase), we will route the request to a high-priority server whose address will be fetched from the identity document.

Add a note hereListing 15-2 shows a sample of an identity document. The XSLT uses the identity document to externalize IP information. This technique could be used for any instance when static data is used with XSLT.

Add a note hereListing 15-2: XML Identity Document with Externalized Routing Information

Add a note here<?xml version="1.0" encoding="UTF-8"?>

<!-- -->

<!-- This document is used to : -->

<!-- Externalize all ip/port information for routing purposes -->

<!-- -->

<identityDocument>

<!-- -->

<!-- Routing info for bookService -->

<!-- -->

<service name="bookService">

<!-- -->

<!-- These addresses will be fetched from within XSLT via xPath -->

<!-- -->

<endPoints>

<BookServiceHost1>

<ip>highPriorityBookHost.books.net</ip>

<port>2129</port>

</BookServiceHost1>

<BookServiceHost2>

<ip>bookHost.books.net</ip>

<port>2130</port>

</BookServiceHost2>

</endPoints>

</service>

</identityDocument>

Add a note hereIn the XSLT that does the actual routing, all that needs to be done is to reference the endpoints within the identity document. The imported identity document is fetched and cached at compilation time, and results in no overhead at execution time. Remember, in many instances routing could have been performed without the use of XSLT; this example uses this technique to demonstrate the identity document principle.

Add a note hereIn the XSLT shown in Listing 15-3, the first step is to parse the identity document from the local: directory via the document() function; this creates a parseable document from which we can use XPath to extract values. Refer to [Chapter 22](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=4078#4078), [“Introduction to DataPower Development”](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=4078#4078) and [Chapter 23](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=4198#4198), [“Programming Within the DataPower Environment”](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=4198#4198) if you need a refresher on XSLT programming within DataPower.

Add a note hereThe document() function actually uses an appended XPath to get only the endpoints. That’s all we are interested in here and will make the following XPath statements a bit simpler. The resultant nodeset is stored in the endPoints XSL variable.

Add a note hereWe check the incoming URL via the system variable var://service/URL-in, and when it contains the Purchase identifier, we route to the host identified in BookServiceHost1 using the xset-target extension element. That is a faster server, and we want to ensure those purchases go through!

Add a note hereListing 15-3: BookServiceRouter.xsl with Inclusion and Reference to Identity Document

Add a note here<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet version="1.0"

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

xmlns:dp="http://www.datapower.com/extensions" extension-element-

prefixes="dp">

<!-- -->

<!-- This stylesheet is used to establish routing -->

<!-- Identity Document, identityDocument.xml, is used for ip/port

resolution -->

<!-- -->

**<xsl:variable name="endPoints"**

**select="document('local:///identityDocument.xml')//identityDocument/service**

**[@name='bookService']/endPoints"/>**

<!-- -->

<xsl:template match="/">

<!-- -->

<xsl:variable name="urlIn" select="dp:variable('var://service/URL-

in')"/>

<!-- -->

<!-- Check the incoming URL, set Route -->

<!-- -->

<xsl:choose>

<!-- -->

<!-- BookService routing -->

<!-- -->

**<xsl:when test="contains($urlIn, 'Purchase')">**

**<!-- -->**

**<!-- Set routing for Purchases -->**

**<!-- -->**

**<dp:xset-target host="$endPoints/BookServiceHost1/ip/text()"**

**port="$endPoints/BookServiceHost1/port/text()" ssl="false()"/>**

**<xsl:message dp:priority="'info'">**

**<xsl:value-of select="concat('Found ', $urlIn, ', Setting**

**routing to ',**

**$endPoints/BookServiceHost1/ip/text(), ':',**

**$endPoints/BookServiceHost1/port/text())"/>**

**</xsl:message>**

**</xsl:when>**

<xsl:when test="contains($urlIn, 'Query')">

<!-- -->

<!-- Set routing for Query -->

<!-- -->

<dp:xset-target host="$endPoints/BookServiceHost2/ip/text()"

port="$endPoints/BookServiceHost2/port/text()" ssl="false()"/>

<xsl:message dp:priority="'info'">

<xsl:value-of select="concat('Found ', $urlIn, ', Setting

routing to ',

$endPoints/BookServiceHost2/ip/text(), ':',

$endPoints/BookServiceHost2/port/text())"/>

</xsl:message>

</xsl:when>

<xsl:otherwise>

<xsl:message dp:priority="'error'">

<xsl:value-of select="concat('Unknown routing based on URL

', $urlIn)"/>

</xsl:message>

</xsl:otherwise>

<!-- -->

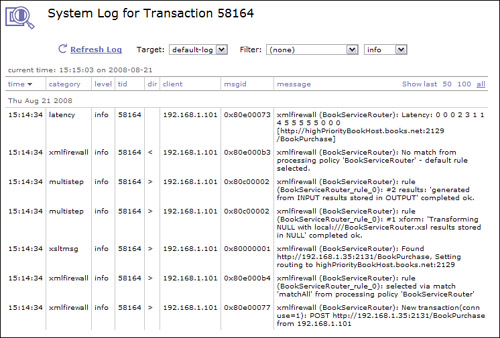
</xsl:choose>

<!-- -->

</xsl:template>

</xsl:stylesheet>

Add a note hereFigure 15-9 shows the results of our routing. The messages we produced are written to the log, and the routing determination is made. Again, the purpose of this example is not so much to present routing as there are many ways to do that (such as using a Route Action) but to demonstrate a simple and effective way to externalize static information from within an XSLT stylesheet. In this manner, the stylesheet could be moved to another environment that contains its own version of the identity document, with different endpoints, and the XSLT will work without change! No environment affinities, no “magic numbers!”

[](javascript:PopImage('IMG_340','http://images.books24x7.com/bookimages/id_30903/15fig09_alt.jpg','746','505'))  
Add a note hereFigure 15-9: Identity document routing log results.

**Note: Do Not Use Magic Numbers (Static Data) in XSLT**

Add a note hereThe use of the identity document principle allows for the construction of environment neutral XSLT. Static data within XSLT forces an affinity to the environment and complicates migration of services.

**Configuring for Migration Summary**

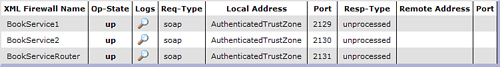
Add a note hereWe have seen how the use of configuration options such as host aliases, DNS static hosts, environment-specific DNS servers, and externalizing magic numbers in XSLT can be effective in easing the issues of configuration migration. Some objects have to be hand-edited when initially establishing a device. For example, each device must have specific Ethernet Interface definitions.

Add a note hereHowever, after a device is established, the need for maintaining these objects is minimal. Your goal in configuration is to not establish an affinity to an environment within service definitions.

**Add a note here****Configuration Migration Tools**

Add a note hereYou’ve spent time configuring your Web Service Proxy and your XML Firewall services. You’re certainly not going to repeat the key strokes on another device. As you might already know, DataPower provides several ways to move configuration data between devices. All three primary configuration methods, WebGUI, CLI, and XML Management, provide options for you. We look at some of the capabilities.

Add a note hereThe following examples utilize a sample domain named BookService, which contains the XML Firewall services shown in Figure 15-10, and the Web Service Proxy shown in Figure 15-11.

[](javascript:PopImage('IMG_341','http://images.books24x7.com/bookimages/id_30903/15fig10_alt.jpg','751','100'))  
Add a note hereFigure 15-10: BookService domain XML Firewall listing.

[Image from book](javascript:PopImage('IMG_342','http://images.books24x7.com/bookimages/id_30903/15fig11_alt.jpg','641','48'))  
Add a note hereFigure 15-11: BookService domain Web Service Proxy listing.

**Package Importing and Exporting**

Add a note hereThere are several methods for exporting and importing configuration details. In addition to the primary configuration methods, configuration modification can be performed by editing the configuration files on the device (with caution), as we will discuss. The on board configuration files located in the config: directory contains a sequential list of CLI commands.

Add a note hereThe methods may well be used in conjunction in a complete configuration migration strategy. For example, an export created by the WebGUI may be referenced by an invocation of the CLI or the XML Management Interface.

**WebGUI Methods**

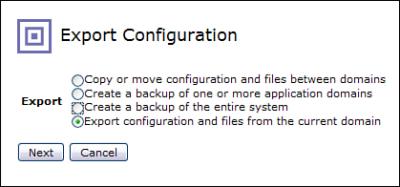
Add a note hereThe simplest way to move configuration between environments is to use the Import/Export Utilities of the WebGUI. This tool allows for the export of configuration details, and allows for the convenient automatic fetching of subordinate objects. For example, if a Web Service Proxy (WSP) is exported and referenced objects are selected, match rules, style policy rules and actions, XML managers, and other supporting objects used by the WSP are exported as well. This technique even fetches XSLT that is identified within actions.

Add a note hereHowever, be aware that this does not pick up XSLT and XML files located on the device’s file system that are included (either by xsl:include, xsl:import or document() functions) by the action’s XSLT. Simply moving local files to a SCM system and fetching from actions or within XSLT eliminates this problem. We address the local file system issue when we talk about configuration architectures, but you need to be aware of it.

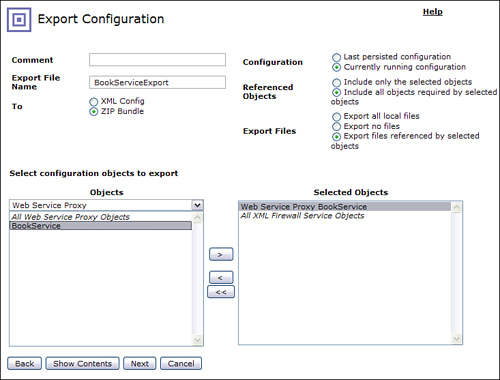
**Tip: XSLT and XML Included by XSLT Are Not Automatically Included in Referenced Objects**

Add a note hereIf you have an action that uses an XSLT and that stylesheet includes other XSLT or XML documents, they are not automatically included among the referenced objects of an export. You must either fetch them from off box repositories or copy them onto the device.

Add a note hereFigure 15-12 shows the options available from the WebGUI export configuration screen. You can create backups of entire application domains, export configuration from the current domain, copy or move configuration between domains, or back up the entire system.

[](javascript:PopImage('IMG_343','http://images.books24x7.com/bookimages/id_30903/15fig12.jpg','419','196'))  
Add a note hereFigure 15-12: Export configuration options.

Add a note hereLet’s look at what is actually exported when we do an export of the aforementioned XML FWs and WSPs. Figure 15-13 shows the export screen after having selected the XML Firewall Service and Web Service Proxy objects. Notice that the Export File Name has been entered (rather than accepting the default export filename), and the To property has been set to ZIP Bundle. This will produce a ZIP document containing the exported objects and files. The XML Config option produces an XML document containing the configuration and exported files in base64 format within XML Elements. Their use is quite similar though the ZIP bundle is compressed.

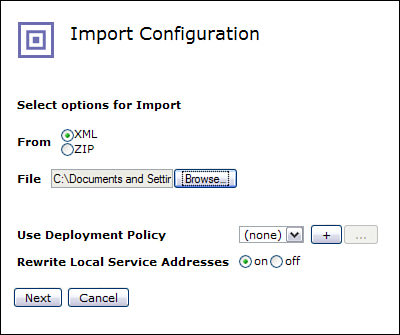
[](javascript:PopImage('IMG_344','http://images.books24x7.com/bookimages/id_30903/15fig13_alt.jpg','733','557'))  
Add a note hereFigure 15-13: Export of BookService domain services using ZIP formatted configuration file.

Add a note hereAs we said earlier, the Export files referenced by the selected objects option does a good job of finding all the objects required by the services. In fact if you want to know ahead of time which objects are to be selected, all you have to do is use the Show Contents button. Figure 15-14 shows the objects associated with our services. Some are obvious, such as the Processing Policy; some are not so obvious, such as the WS-Proxy Endpoint Rewrite rules. These are hidden children of the WSP, and ones that you might not even be aware of. They are associated with the remote and local properties of WSP endpoints. The point here is that you cannot just assume that an object lives by itself, the objects have subordinates and some are not readily apparent.

[](javascript:PopImage('IMG_345','http://images.books24x7.com/bookimages/id_30903/15fig14_alt.jpg','521','632'))  
Add a note hereFigure 15-14: Manifest of objects selected for export.

Add a note hereAlso notice that the private keys are identified as dependents of the services but are not exported. This is a fundamental feature of the appliance. It will not release a private key or certificate from the private certificate directories (cert: and sharedcert:); this is a valuable security attribute. If you have read [Chapter 18](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=3288#3288), [“DataPower and SSL,”](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=3288#3288) then you understand that when you create keys and certificates on the device (using Crypto Tools), you are given a one-time opportunity to save off the keys in the temporary: directory. They may be copied off the device at this time. Be forewarned though, the temporary: directory is cleared on every device reboot, though not on a domain restart or firmware reload.

Add a note hereImporting packages via the WebGUI is a simple and straightforward process as seen in Figure 15-15. Simply use the Import facility, and select the exported ZIP or XML bundle. You’ll notice a couple of options here also.

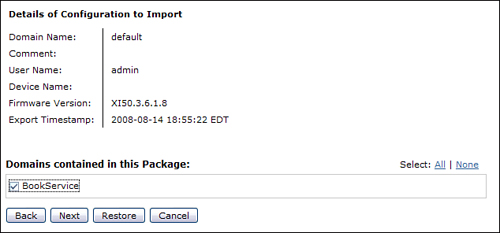
  
Add a note hereFigure 15-15: Import configuration with configuration modification options.

Add a note hereThe first option is Use Deployment Policy. We explore this feature shortly; it allows us to build a powerful filtering and selection mechanism to determine which objects from the export bundle are actually imported. It enables us to dynamically change some of the values of properties as they are imported.

Add a note hereThe next option is Rewrite Local Service Addresses. This option works with the Ethernet addresses assigned to objects such as XML Firewall and FSHs. Remember we have stressed in most cases it is a best practice to use host alias objects and not the actual address of an interface. If you need to refresh your understanding of interfaces on DataPower, you can refer to [Chapter 4](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=403#403).

Add a note hereThis feature works only on raw physical addresses. If you had assigned the address of eth1 (whatever the dot decimal IP address is) to an XMLFW, this feature would change that value to the dot decimal of eth1 on the destination device.

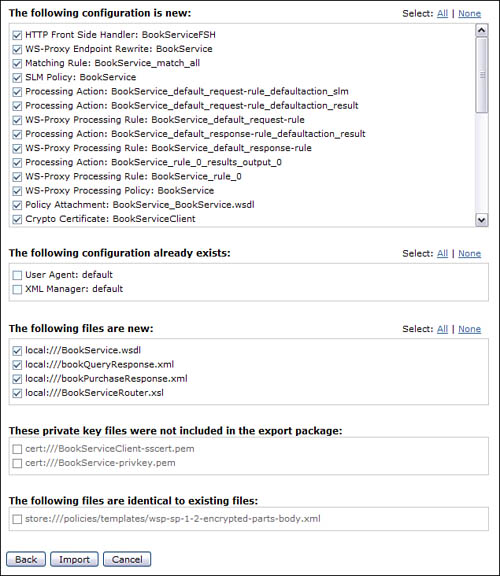
Add a note hereIf the export contained multiple domains, you would be given the choice of which domains to import. If you attempt to import a configuration exported from a domain into the default domain, you would be warned as application domains are normally imported into application domains. Figure 15-16 shows the selection of the BookService domain configuration

[](javascript:PopImage('IMG_347','http://images.books24x7.com/bookimages/id_30903/15fig16_alt.jpg','634','296'))  
Add a note hereFigure 15-16: Import domain selection.

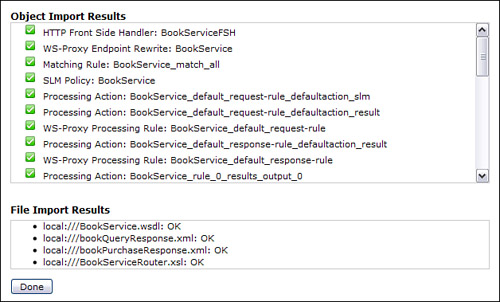
**Tip: Avoid Domain Exports When Migrating Services from Development Domains**

Add a note hereWhen migrating services from a development domain where many unrelated configuration objects may exist, it is better to export the individual service objects and their referenced objects rather than the entire domain. In this manner, you have greater control over which objects are actually exported and eventually imported into the target domain.

Add a note hereFigure 15-17 shows all the objects that were exported from the source platform. You can selectively import individual objects from the list if you desire by checking or unchecking the box next to the object. You will be alerted if the objects are already on the target platform, or if they are identical to the objects on the target platform. Notice that the keys and certificates are identified, but no option to import is provided. A file from the store: directory is also identified without an import option. Recall from the device administration discussions that you may not modify the contents of the store: directory unless you are the device administrator and you are importing into the default domain.

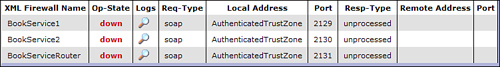
[](javascript:PopImage('IMG_348','http://images.books24x7.com/bookimages/id_30903/15fig17_alt.jpg','635','732'))  
Add a note hereFigure 15-17: Import configuration detail with optional and unexported objects.

Add a note hereFinally, the success or failure of the importation process is described. Any errors would be presented. As you can see in Figure 15-18, all is well.

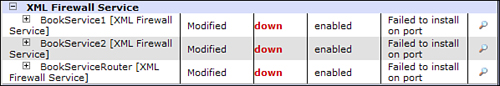
[](javascript:PopImage('IMG_349','http://images.books24x7.com/bookimages/id_30903/15fig18_alt.jpg','628','379'))  
Add a note hereFigure 15-18: Import results.

Add a note hereWe have now completed our task. We imported the three XML Firewalls, the Web Service Proxy, and their supporting objects onto the destination device. We are ready to utilize our imported configuration. Or are we? We need to look at the services on the target device and make sure they are up and enabled.

Add a note hereFigure 15-19 shows the status of the Firewalls. Notice that they are in Op-State down! How could that be, when we received success messages on the import?

[](javascript:PopImage('IMG_350','http://images.books24x7.com/bookimages/id_30903/15fig19_alt.jpg','751','100'))  
Add a note hereFigure 15-19: XML Firewalls in down op-state after import of configuration.

Add a note hereLooking closer at the service’s status from Status→Object Status shows that they have failed to install on the port; you can see that in Figure 15-20. This is typically caused when another service is already assigned the port the service is attempting to use. This problem is encountered when environments do not share network details. We will see shortly how this problem can be addressed quite easily.

[](javascript:PopImage('IMG_351','http://images.books24x7.com/bookimages/id_30903/15fig20_alt.jpg','598','103'))  
Add a note hereFigure 15-20: XML Firewalls object status.

Add a note hereWe have seen that although the majority of configuration details can be easily and confidently copied across devices, there are certain properties that require a little more attention. As in our example, the ports used in a test environment might be different from those used in a production environment.

Add a note hereFortunately, the Deployment Policy can be used to filter which objects are allowed to be imported, and it can change properties of objects that are imported. Let’s look at how it works. First, Figure 15-21 shows the use of the deployment policy on the Import Configuration screen.

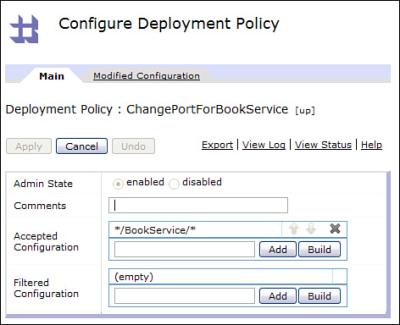
[](javascript:PopImage('IMG_352','http://images.books24x7.com/bookimages/id_30903/15fig21_alt.jpg','513','262'))  
Add a note hereFigure 15-21: Designation of deployment policy on the import of a configuration.

Add a note hereOpening up the deployment policy object (as always via the ... button), shows some of the configuration options. The deployment policy has two primary objectives.

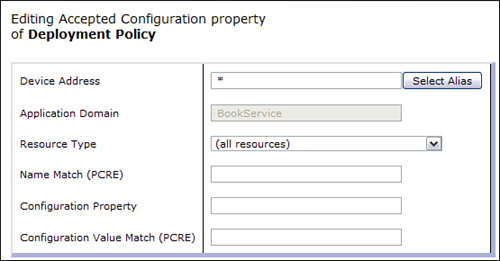
Add a note hereThe first is to select which objects are allowed into the target device. This is done by the definition of a white-list (acceptable objects) and a black-list (filtered objects). These define a fine-grained selection list. The white-list could be all XMLFW objects while the black-list could be a single XMLFW named dont-import-me. The white-list is not required and if not used, all objects will be accepted unless they are contained within the white-list specification. The black-list is not required, and only those objects within the black-list range that are not to be imported need be identified.

Add a note hereThe second objective is to modify properties of certain objects. You define the selection criteria and the properties to modify. You can add new values, change values, and remove them.

Add a note hereOur example is just accepting anything that came from the BookService domain. Figure 15-22 shows the configuration. Multiple URIs could have been entered allowing for an expanded white-list, and we could have also configured Filtered Configurations.

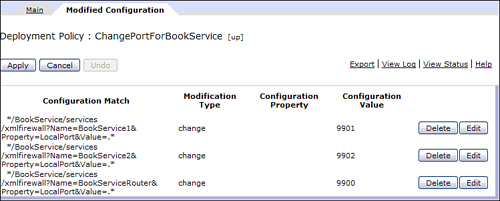
[](javascript:PopImage('IMG_353','http://images.books24x7.com/bookimages/id_30903/15fig22.jpg','472','384'))  
Add a note hereFigure 15-22: Deployment Policy accepted configuration.

Add a note hereThe Build button allows for the easy creation of the acceptance and filter configuration strings. As you can see in Figure 15-23, selection can be based on a variety of properties such as the original device address and domain, resource type and name, and even the value of a particular property the object might contain. Again, here we are just allowing any object from the BookService domain.

[](javascript:PopImage('IMG_354','http://images.books24x7.com/bookimages/id_30903/15fig23_alt.jpg','555','290'))  
Add a note hereFigure 15-23: Deployment Policy accepted configuration build.

Add a note hereNow that we have identified the objects to allow, it’s time to get to what we are really after, modifying properties as they are imported. This detail is defined within the Modified Configuration tab.

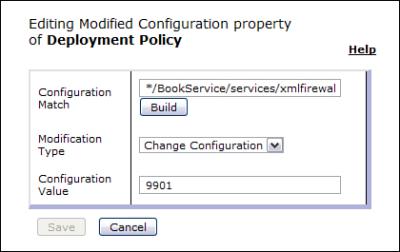
Add a note hereAs you can see in Figure 15-24, it allows multiple entries, each identifying a particular property to be modified. You can use a wide brush via wildcards, or be selective and just identify a particular object. Some of the properties (Name Match, for example) accept PCRE expressions such as ‘.\*’, whereas others (such as Application Domain) accept ‘\*’ as the wildcard specification. The build button makes these uses clear.

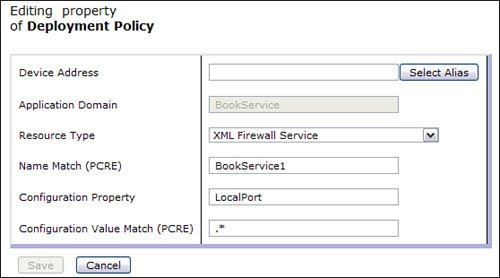
[](javascript:PopImage('IMG_355','http://images.books24x7.com/bookimages/id_30903/15fig24_alt.jpg','749','301'))  
Add a note hereFigure 15-24: Deployment Policy modified configuration.

Add a note hereWe’ll look at the details shortly, but these entries will select any XML Firewall with a name of BookService1, BookService2, and BookServiceRouter that has a LocalPort property of any value (.\*), and then selectively change the firewall’s port. This allows us to just change the individual properties that vary between the export and import environments.

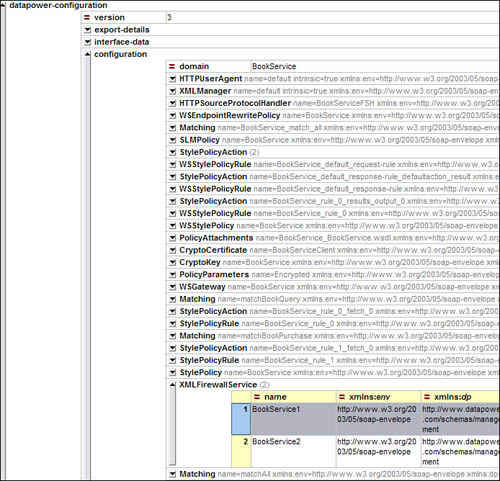
Add a note hereEach entry within the Modified Configuration contains the Configuration Match (again identifying the object and property), and a Modification Type that allows you to change the value, add additional values, or remove the matched value. Finally, the Configuration Value contains (in this case, as it’s a change) the new property value to use. Figure 15-25 shows the completed configuration property assigning a new port value of 9901 to BookService1. (This is at the end of the string and not visible in this display, but was shown in Figure 15-24.)

Add a note hereYou may have been intimidated by the configuration match string, but fear not, the Build button (seen in Figure 15-25) allows for easy creation, just as with the accepted and filtered configuration URI strings. All you have to do is identify the resource—there’s even a drop-down box for the resource type (objects), and you can use PCRE expressions to match multiple objects with a single string. In Figure 15-26, the XML Firewall Service objects named BookService1 from the BookService domain that have a LocalPort property (with any value) are selected.

[](javascript:PopImage('IMG_356','http://images.books24x7.com/bookimages/id_30903/15fig25.jpg','420','265'))  
Add a note hereFigure 15-25: Deployment Policy change value.

[](javascript:PopImage('IMG_357','http://images.books24x7.com/bookimages/id_30903/15fig26_alt.jpg','558','310'))  
Add a note hereFigure 15-26: Deployment Policy configuration match builder.

Add a note hereNow the cautious reader might ask, “How did you know the property name was LocalPort?” That is a good question, because there is no documented listing of property names. To find the name of the property requires investigation. Remember when we exported the BookService services to a zip file named BookServiceExport.ZIP? Well, it contains an XML File named export.xml that has all the details of the export. As you can see in Figure 15-27, each of the exported objects is contained within. In fact the two firewalls are clearly shown.

[](javascript:PopImage('IMG_358','http://images.books24x7.com/bookimages/id_30903/15fig27_alt.jpg','694','668'))  
Add a note hereFigure 15-27: XML details of a configuration export.

Add a note hereListing 15-4 is an expanded view of the configuration for BookService1. Only the first few elements are displayed. However, as you can see, the LocalPort is shown as containing the value of 2129. Now the reality is that you will not be changing many properties, and after you do this once or twice, you will be comfortable with it.

Add a note hereListing 15-4: XML Details of Exported XML Firewall Configuration with LocalPort Highlighted

Add a note here<XMLFirewallService name="BookService1"

xmlns:env="http://www.w3.org/2003/05/soap-envelope"

xmlns:dp="http://www.datapower.com/schemas/management">

<mAdminState>enabled</mAdminState>

<LocalAddress>InternalTraffic</LocalAddress>

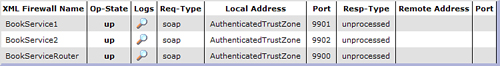
<UserSummary>an example XML Firewall Service</UserSummary>

<Priority>normal</Priority>

**<LocalPort>2129</LocalPort>**

<HTTPTimeout>120</HTTPTimeout>

Add a note hereExecuting the import now with the deployment policy performs the changes we desired. The ports have been changed. This is just what we wanted! Figure 15-28 shows the objects in the up Op-State.

[](javascript:PopImage('IMG_359','http://images.books24x7.com/bookimages/id_30903/15fig28_alt.jpg','750','99'))  
Add a note hereFigure 15-28: Firewalls in up op-state after importing with deployment policy.

**CLI Methods**

Add a note hereThe CLI has the advantage of providing for automated, scripted execution of Import/Export functions. Importantly though, not all the functionality of the WebGUI is supported. However, the CLI has a special place in the pantheon of configuration methodologies. It is the format of the on-board configuration file as was seen earlier in our discussion of the configuration file format and in the example of application domain definition (way back in Listing 15-1).

Add a note hereWe’ll be especially interested in CLI functionality that leverages the more-advanced features of the other methods such as creating a selective export of services via the WebGUI. This is important, because as you will see, we can use these commands within the device’s configuration file to fetch configurations and files and to import previously exported packages.

**Including Configuration**

Add a note hereThere is no need to have the device’s entire configuration details resident on the device. In fact, fetching and sharing configuration resources with other devices is a key component of modular configuration design. The CLI can be used to include source configuration from its file system or external sources using protocols such as HTTP, HTTPS, FTP, or NFS. Secured protocols such as HTTPS should be used in unsecured environments to protect configurations while in transit.

Add a note hereThe CLI Exec command is used for this purpose, as seen in Listing 15-5; this causes the included configuration to be executed as if it were originally written in the configuration file. This command could be executed from an interactive CLI session, but it is even more useful when it is part of the device’s boot configuration file, and by virtue of its execution, the startup configuration fetches modularized configuration components.

Add a note hereListing 15-5: Inclusion of External Configuration via “exec” Command

Add a note hereexec http://192.168.1.35:8080/SharedBookService.cfg

**Exporting Configuration**

Add a note hereThe CLI export functions are performed via the backup command. This functionality is equivalent to the backup entire domain capability of the WebGUI and does not support the selective filtering of objects. However as will be seen shortly, the CLI can be used to import a selective export created by the WebGUI or XML Management Interface.

Add a note hereWhen executed within the default domain, the Backup command can select which domain to backup via an optional domain argument. When executed in an application domain, only the current domain is exported.

Add a note hereListing 15-6 shows the execution of the Backup command. Notice that the provided filename (which is the same as the domain name in this example) is appended with the .zip suffix. The XML format is not available here.

Add a note hereListing 15-6: Backup with Domain

Add a note herexi50[BookService](config)# backup BookService

Backup to 'BookService.zip' scheduled (may take a few minutes to complete).

Backup is complete.

**Importing Packages**

Add a note hereThe CLI supports importing of packages created by all export methods. This process is performed by first creating an Import Package describing the export and defining import options. The package is then processed or executed to invoke the import package process.

Add a note hereSeveral options such as turning on/off overwriting of files and objects are supported, as is automatically performing the import at device startup via the auto-execute property. Deployment policies can also be associated with the package so that when they are imported, all the changes described in the policy will be made.

Add a note hereThe import-exec command is used to perform an import of a previously exported configuration package. Listing 15-7 shows the import-package configuration. This is the CLI command structure used to define the import-package.

Add a note hereListing 15-7: Import Package Configuration with Deployment Policy

Add a note hereimport-package "bookService"

source-url "https://192.168.1.35:8080/BookServiceExport.zip"

deployment-policy ChangePortForBookService

no auto-execute

exit

Add a note hereListing 15-8 shows two command line CLI executions. The first does a show command to list out the bookService package. This allows for the confirmation of option defaults which were not set in the package creation. You can see that the overwrite and rewrite local IP options are set to on.

Add a note hereThe second command, import-exec, demonstrates the actual execution of the import process. As with the previous exec command, the creation of the import package and the import-exec command can be performed in an interactive CLI session as in the example in Listing 15-8. They could also be part of the device’s boot configuration.

Add a note hereListing 15-8: Details of Import-Package Configuration Object and Execution of Package

Add a note herexi50[BookService](config)# show import-package bookService

import-package: bookService [up]

---------------------------

admin-state enabled

source-url https://192.168.1.35:8080/BookServiceExport.zip

import-format ZIP

overwrite-files on

overwrite-objects on

deployment-policy ChangePortForBookService [up]

local-ip-rewrite on

auto-execute off

xi50[BookService](config)#

xi50[BookService](config)# import-exec bookService

Loading import-package 'bookService'.

Import package is complete.

**Including Files**

Add a note hereAlthough the previous discussion has dealt with configuration packages, there might be instances where individual files (XSLT or XML) need to be imported into the device at startup. Recall when we originally brought this subject up; this is unnecessary if you fetch files from external sources such as an SCM repository.

Add a note hereIn either case, this is easily performed with the copy command. Listing 15-9 demonstrates the copy command format. The -f option forces a rewrite if the file already exists on the target platform. Again, all supported protocols can be used, and you should use secured protocols in a production environment. As with all CLI commands we have seen, the copy command can be executed interactively or as part of the boot configuration. This technique may also be used to import key and certificate files from the SCM into the cert: directory of a domain.

Add a note hereListing 15-9: Execution of Copy Command to Fetch External File

Add a note herecopy -f https://192.168.1.35:8080/IdentityDocument.xml

local:///IdentityDocument.xml;

**Add a note here****XML Management Methods**

Add a note hereThe XML Management interface is a valuable tool in the management of configurations. It supports a similar set of commands as the WebGUI and CLI. Packages can be exported, imported with deployment polices, configuration files can be loaded onto the device, and the device can be restarted.

Add a note hereListing 15-10 shows the formatting of a do-import command. This is analogous to the CLI’s import-exec command, or the Import feature of the WebGUI. Notice that a deployment policy is defined via an attribute on the dp:do-import element, and that the dp:input-file element contains the actual import package. It needs to be base64 encoded first, and the actual data has been truncated from this display for brevity. Tools such as OpenSSL ([www.openssl.org](http://www.openssl.org)) can be used to perform the base64 encoding, or if you are writing XSLT, you can use the DataPower encode() function.

Add a note hereListing 15-10: XML Management Do-Import

Add a note here<?xml version="1.0" encoding="UTF-8"?>

<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">

<env:Body>

<dp:request domain="bookService"

xmlns:dp="http://www.datapower.com/schemas/management">

**<dp:do-import source-type="ZIP" deployment-**

**policy="changeHostAliasAndIPAddress">**

<dp:input-file>

**{base64 input}**

</dp:input-file>

</dp:do-import>

</dp:request>

</env:Body>

</env:Envelope>

Add a note hereAs we will see, the XML Management interface will play an important role in device management. Among the primary reasons is the ability to script its execution using tools such as ANT. Doing so provides a platform for configuration modification, migration, execution, and device administration.

Add a note hereYou should read the WebGUI product guide for definitive information on XML Management request structure and execution.

**Add a note here****Configuration Structure for High Availability and Consistency**

Add a note hereNow that we have discussed configuration best practices and methods for exporting and importing configuration details, we can demonstrate methods for their utilization in DataPower device configuration management. This configuration will utilize a three-tiered approach to device management: a device specific fixed component, a component which is shared across devices within an environment, and the support for application domains. The objectives are as follows:

* Add a note hereProvide for a consistent configuration across devices within an environment.
* Add a note hereProvide for a consistent configuration through device restart.
* Add a note hereUtilize Source Control for configurations.
* Add a note hereSegregate environment-specific from shared and application domain configurations.
* Add a note hereProvide a platform for service configuration migration and promotion.

**Device Configuration**

Add a note hereEach device contains a section of its configuration that is unique from all other devices. At a minimum this is the definition of its Ethernet Interfaces but may contain other objects such as host aliases. No other configuration can be fetched from external resources without first configuring the interfaces!

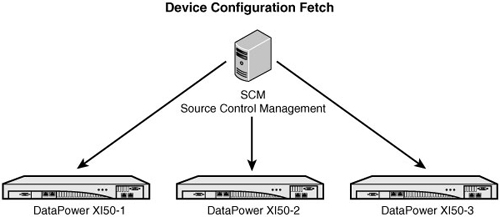
Add a note hereEach device shares many configuration details with other devices in its environment. DNS and Network Time Protocol (NTP) servers, for example, are all used with similar configuration details. Administrative objects such as user-groups, logging targets, access control lists, and document caching policies will also typically be shared across all devices in an environment. In addition, the application domains are configured in the default domain and are shared across devices.

Add a note hereServices should always be configured in application domains. Services consist of the services themselves and their supporting style policies, rules, actions, AAA policies, match rules, and XSLT, among other objects. It is the services that will bear the majority of ongoing maintenance and, as such, we need to make special provisions for their definition.

**Tip: Always Configure Services in Application Domains**

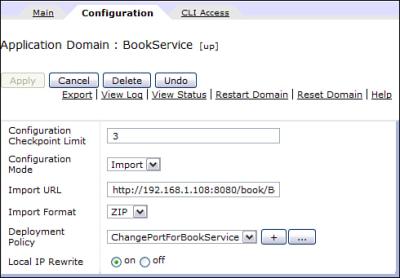
Add a note hereConfiguring services in application domains (rather than the default domain) provides far greater control in access and management. Access privileges may be limited by domain providing administrative restrictions, and many device configuration options are designed to leverage the domain architecture. The default domain should only be used for device configuration.

Add a note hereRather than having the service configuration reside on the device, it is desirous that it be fetched from an external source, and furthermore that this resource be under the control of an SCM. Figure 15-29 demonstrates graphically the concept of configuration fetch.

[](javascript:PopImage('IMG_360','http://images.books24x7.com/bookimages/id_30903/15fig29_alt.jpg','600','260'))  
Add a note hereFigure 15-29: Configuration fetch from SCM.

Add a note hereIn the following examples, techniques that can be employed to achieve this goal are demonstrated. By using these techniques, the device refreshes itself to a known good configuration at each controlled restart, or even in the event of an uncontrolled restart.

Add a note hereThe example in Figure 15-30 illustrates fetching application domain configuration details. The Import URL represents an SCM location of a previously exported configuration. Notice that the deployment policy is also supported here.

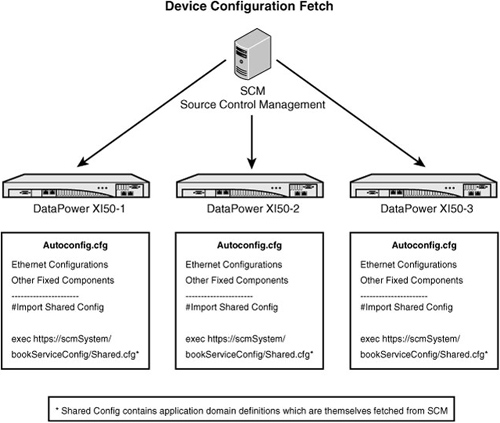
[](javascript:PopImage('IMG_361','http://images.books24x7.com/bookimages/id_30903/15fig30.jpg','490','341'))  
Add a note hereFigure 15-30: Domain configuration with imported configuration and deployment policy.

**Tip: Always Use Secured Protocols for Fetching Configuration Data**

Add a note hereAlthough the examples shown in Figure 15-30 show the use of HTTP, any production system should always use secured protocols such as HTTPS to protect configuration details from unscrupulous intermediaries.

**Sample Device Configuration**

Add a note hereThere is no definitive methodology for configuration architecture. Figure 15-31 expands on our configuration fetch graphic of Figure 15-30 by demonstrating an option that provides for several desirable features. First, by relying on an SCM system as the repository for configuration, control and audit are provided. Second, modularity of the configuration is enhanced by separating device specific data from shared and service specific details.

[](javascript:PopImage('IMG_362','http://images.books24x7.com/bookimages/id_30903/15fig31_alt.jpg','640','541'))  
Add a note hereFigure 15-31: Configuration architecture with autoconfig.cfg.

Add a note hereIn this scenario, each device contains a unique autoconfig.cfg configuration file. It defines the device specific properties and then fetches the shared configuration which itself contains (among other objects) the application domain definitions which then fetch their configuration from the SCM!

**Caution When Editing Device Configuration File**

Add a note hereCare must be exercised when directly editing the onboard device configuration file. Specifically, you want to make sure you maintain the Ethernet interfaces. If you do not configure the interfaces, you will have no TCP access to the device! Similarly, make sure you define the WebGUI, SSH, XML Management Interface, and other configuration methods you intend to use. You can always get the current configuration by viewing or downloading the autoconfig.cfg file from the config: directory of the default domain. If you do not define the Ethernet interfaces, you can still get access to the device via the serial port. Please refer to the WebGUI Guide for your device for additional information.

**Warning: Use Caution When Directly Editing Device Configuration Files!**

Add a note hereYou should have serial cable access as a backup. This does not require TCP access and can be used to access an unconfigured (or incorrectly configured) device.

Add a note hereBefore you maintain the device configuration file you should

* Add a note hereMake an entire device backup via the WebGUI Export Utility.
* Add a note hereCopy the current autoconfig.cfg to another file in the device’s config: directory.
* Add a note hereHave a serial cable and physical access to the device as a fail-safe plan for device access.

**Fixed Component of config:///autoconfig.cfg**

Add a note hereThe example shown in Listing 15-11 demonstrates a sample bare-bones default domain configuration file, which could be uploaded to the config: directory of the device and identified as the boot config on the System Control Panel. It defines the Ethernet interface, host aliases, system contact info, and the WebGUI and the XML Management Interfaces.

Add a note hereIt is important to note this represents an example of a fixed configuration file. Your implementation should be based on your particular requirements. The easiest way to define this configuration is to create the necessary objects via the WebGUI and to download the autoconfig.cfg file, make the necessary changes, and to then upload the file back onto the config: directory.

Add a note hereThe configuration executes (via exec command) an off box configuration that contains configuration information shared by all devices in the environment. This shared configuration (shown shortly) contains the Application Domain definitions.

Add a note hereListing 15-11: Sample Default Domain Fixed Config

Add a note hereconfigure terminal

# This is the fixed component of the 'Default' Domain for DataPower device

interface "eth0"

ip address 192.168.1.35/24

mtu 1500

ip default-gateway 192.168.0.1

arp

mode 1000baseTx-FD

exit

interface "eth1"

ip address 192.168.1.36/24

mtu 1500

ip default-gateway 192.168.0.1

arp

mode 1000baseTx-FD

exit

system

contact "bookservice admin"

name "prod bookservice Alpha"

location "4th Floor Networking"

exit

host-alias "AuthenticatedTrustZone"

reset

ip-address 192.168.1.35

exit

host-alias "AuthenticatedAdminZone"

reset

ip-address 192.168.1.36

exit

# Web and XML-MGMT Interfaces can be restricted to a INT, # so put them in

Fixed Config

web-mgmt

admin-state enabled

local-address AuthenticatedAdminZone 9090

idle-timeout 6000

exit

xml-mgmt

admin-state enabled

local-address AuthenticatedAdminZone 5050

mode any+soma+v2004+amp+slm

exit

# Now pull in the variable part of the configuration,

# this is the same for all

# devices in this environment, i.e. Dev/Test/Prod

# This contains the application domain definitions

exec https://192.168.1.35:8080/SharedBookService.cfg

**Shared Default Domain Configuration, Including Application Domain Definitions**

Add a note hereThe next example, shown in Listing 15-12, demonstrates a sample variable component of the default domain. It configures objects such as DNS, Simple Network Management Protocol (SNMP), and NTP that are shared by all devices in the environment. It also contains the definition of the bookService application domain that itself identifies an off-box location for its configuration and uses a Deployment Policy.

Add a note hereIt is important to note this represents an example of a shared configuration file. Your implementation should be based on your particular requirements. The easiest way to define this configuration is to extract the shared components out of the autoconfig.cfg file downloaded from the config: directory.

Add a note hereListing 15-12: Sample Default Domain Shared Config

Add a note here# This is the variable component of the 'Default' Domain

# for all DataPower devices in an environment

dns

admin-state enabled

search-domain "some.com"

name-server 152.155.21.10 53 53 0 3

name-server 152.155.21.60 53 53 0 3

exit

alias "reload" "flash;boot config autoconfig.cfg;shutdown reload"

ntp-service

admin-state disabled

remote-server 152.159.20.10

remote-server 152.159.20.60

exit

timezone EST5EDT

snmp

admin-state enabled

ip-address 192.168.0.52

community "public" "default" "read-only" "AuthenticatedTrustZone"

exit

ssh AuthenticatedAdminZone 22

save-config overwrite

usergroup "BookService"

summary "Book Service Admin"

access-policy \*/BookService/\*?Access=r+w+a+d+x

access-policy \*/default/\*?Access=r

exit

deployment-policy "ChangePortForBookService"

accept \*/BookService/\*

modify

"\*/BookService/services/xmlfirewall?Name=BookService1&Property=LocalPort

&Value=.\*" "change" "LocalPort" "9901"

modify

"\*/BookService/services/xmlfirewall?Name=BookService2&Property=LocalPort

&Value=.\*" "change" "LocalPort" "9902"

modify

"\*/BookService/services/xmlfirewall?Name=BookServiceRouter&Property=

LocalPort&Value=.\*" "change" "LocalPort" "9900"

exit

domain "BookService"

base-dir BookService:

base-dir local:

config-file BookService.cfg

visible-domain default

url-permissions "http+https+snmp+ftp+mailto+mq"

file-permissions "CopyFrom+CopyTo+Delete+Display+Exec+Subdir"

file-monitoring ""

config-mode import

**import-url "https://192.168.1.09:8080/BookService.zip"**

**deployment-policy changePortForBookService**

exit

**Service Promotion**

Add a note hereService promotion describes the migration of newly developed configuration specifications (typically services) through the Software Development Life Cycle (SDLC). This topic is vital for the control of configuration assets and availability of production systems. Changes to configuration must not cause service interruptions, and the configuration architecture must provide for a resilient, failsafe platform that is able to reconstruct itself to a known good synchronization point in the event of unexpected system restarts.

**The Environments**

Add a note hereThe development environment is a no-holds-barred arena where new ideas may be developed and tested without the risk of hindering production systems or the functionality of other services. In the DataPower environment, domains may be created for special purposes, and individual services may be altered and their functionality unit tested without respect to other services that may exist on the device. In this environment, unfortunately often little attention is given to security restriction within the application domains because doing so may hinder creativity and productivity.

Add a note hereThe test environment is a more stable arena. Its purpose is to validate configurations. This includes established, as well as newly modified, services because a change to one service may have an unintended effect on another. A testing mechanism includes a collection of scripted service interactions that validate all methods practical and compare those results to a collection of expected results. Many testing methodologies exist, and they may range from simple batch files with request and result compares, to the implementation of complex testing packages. In the test environment, security is critical. No one should be able to modify service configurations once they are established within the test environment so that the testing mechanism itself doesn’t become compromised.

Add a note hereThe acceptance environment serves two purposes. First, it serves as a validation mechanism across the entire device. Here all services are tested and validated. This ensures that not only are all the services within a domain functioning as required, but that no cross-domain dependencies have been affected. However, the acceptance environment’s utility is truly in the validation of the mechanism in which the production system will ultimately be configured, and how it will fetch its configuration and respond to system events. As with the test environment, no one should be able to modify service configurations once they are established within the acceptance environment.

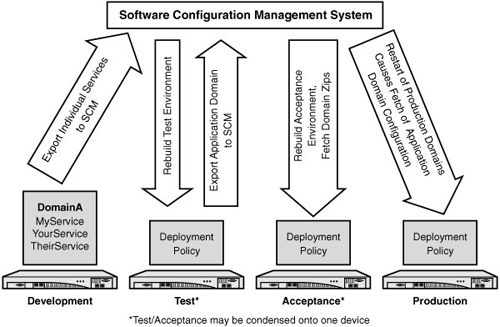
Add a note hereThe production environment is of course why all the previous environments exist. The configuration method must ensure that the previously validated domain configurations are properly reconstructed in the event of a system event. This may be performed by utilizing DataPower configuration techniques, such as application domain definitions which fetch their configuration from external sources at startup, and inclusion of external configuration. Of course, no one should make application domain-level configuration changes on a production device.

**Environmental Differences**

Add a note hereIn an ideal infrastructure, there would be no differences in the configuration of services across the SDLC environments. For example, the IP address of backend applications and identification repositories would be the same, queue names and other aspects of MQ topography would be consistent, and ports used for front side service requests would be consistent. Fortunately, DataPower has provided many mediating strategies to alleviate these differences. We have presented Host Aliases, Static Hosts, DNS, and other techniques that make these differences manageable. And we have also discussed the Deployment Policy and how it can be used to change configuration details during the configuration import process. Using these tools in conjunction with the export and import capabilities inherent in the device should make the vast majority of environmental differences manageable.

**Application Promotion in Detail**

Add a note hereFigure 15-32 demonstrates a complete promotion strategy from development through production. The Software Configuration Management (SCM) System plays an integral part in the facilitation of a secured and auditable configuration repository.

[](javascript:PopImage('IMG_363','http://images.books24x7.com/bookimages/id_30903/15fig32_alt.jpg','600','392'))  
Add a note hereFigure 15-32: Application life cycle utilizing SCM and deployment policies for affinity resolution.

Add a note hereIn the development environment, individual services are exported to the SCM upon successful completion of unit testing. This migration could be performed by the developer through the WebGUI export utility, or it may be part of an automated, scripted process that utilizes the XML-Management interface. In either case, the end result is an export that will be stored within the SCM. This technique serves two necessary functions; not only is an archival recording of the service configuration maintained, but the export of the individual service avoids the potential problem of exporting orphaned objects and other unnecessary configuration details that may be resident in the development domains. Had the entire domain been exported, these ancillary objects would have been part of the export, potentially corrupting the test, acceptance, and production environments.

Add a note herePreparation for testing within the test environment is an automated, scripted process. First, the target domain is deleted from the device. This ensures that no residual objects remain that may effect testing either in a false positive or negative fashion. Second, all services associated with the target domain are imported from the SCM. These are the exports created from the development environment merged with the previous generation of the application domain configuration. This process will utilize a domain manifest that describes the individual services comprising a domain. Deployment policies are utilized to manage environmental differences. Upon completion of the domain-wide tests, the entire domain is exported to the SCM. There is no fear of orphaned objects in this export. This is a fresh domain that contains only the services that are required.

Add a note hereThe acceptance environment is now established for complete device-level testing and production promotion validation. Once again, the domain is reconstructed via the scripted XML-Management interface, establishing a remote configuration with an Import URL pointing to the domain export that was created from the test environment. A deployment policy is established and utilized to address environmental differences. A restart of the domain causes the device to fetch the domain configuration from the SCM, and the domain is ready for acceptance testing.

Add a note herePromotion to production is similar to that of acceptance. The domain is reconstructed and refreshed from the remote SCM repository, and deployment policies are used to address environmental differences. It is important to note that post-production verification must be performed to ensure that all went well before releasing the newly configured device.

Add a note hereThe reader is encouraged to refer to a two-part IBM developerWorks article related to these topics: “Managing WebSphere DataPower SOA Appliance Configurations for High Availability, Consistency, and Control.” Part one contains information relevant to the topics presented in this chapter. In addition, part two contains an example configuration complete with ANT scripts and executions of the XML-Management features required for an automated application promotion methodology.

**Add a note here****Use of External Tools in Configuration Management**

Add a note hereIBM provides several tools that assist in device management. For example, WAS version 7.0 manages multiple devices simultaneously. A single device can be managed to act as the configuration master for other devices. The methods we’ve discussed here still apply. However, these tools then distribute the configuration across the dependent devices. Please see [Chapter 29](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=5160#5160), [“Multiple Device Management Tools”](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=5160#5160) for more information on the use of these tools.

## Summary

Add a note hereThere are many way to manage the configuration of the DataPower device. We demonstrated some basic principles that provide for a consistent configuration across device restarts and allow for a service promotion methodology. However, just as importantly, we discussed the significance of configuring a device in an environment neutral fashion, thereby avoiding problems that might be encountered during migration. Adherence to these principles will save you from having to address these issues later.